Assignment 6 CS 532 Yuhan Xu

1. A is $^{n\times n}$ square symmetric and rank 1 matrix with right singular vectors v_k.

$A = \sum_i \sigma_i u_i v_i^T$

$B = A^TA = (\sum_j \sigma_j v_j u_j^T)(\sum_i \sigma_i u_i v_i^T)$

$= \sum_{i,j} \sigma_i \sigma_j v_j (u_j^T u_i) v_i^T = \sum_i \sigma_i^2 v_i v_i^T$

$B^k = \sum_i \sigma_i^{2k} v_i v_i^T$

when $k \to \infty$, for $i > 1$ $\frac{\sigma_i^{2k}}{\sigma_1^{2k}} \to 0$

$\Rightarrow B^k \approx \sigma_1^{2k} v_1 v_1^T$ for each $i > 1$, $\sigma_i(A) < \sigma_1(A)$

$\Rightarrow$ the power method converges to v_1.

$\left|\frac{\lambda_2}{\lambda_1}\right| \le 1\%$

Since $\lambda_2 = 0$, one iteration is needed to converge within 1% of v_1.

2.

a) No. By using the rotate tool, it seems that a 2-d plane cannot include all the data.

b) use principal component analysis

c) No.

d) a one-dimensional subspace does not capture the data very well, because when I rotated the figure I found the data are in a two-dimensional subspace.

e) $x_{zi} = v_i s u_i^T$. If a is unit-norm vector represent the best one-dimensional subspace for the data, $U = u_1$, $S = s_1$, $V^T = v_1^T$. Since $x_{zi} \approx a w_i$, it means $v_1 s_1 u_1^T = a w_i = v_1 w_i$. So $w_i = s_1 u_1^T$.

f) $b = \frac{\sum_{i=1}^{1000} x_i}{1000}$ which is the averge of mean distance.

g) $X = u_1 s_{11} v_1^T$. $E = u_2 s_{22} v_2^T + u_3 s_{33} v_3^T$. So $\|E\|_F^2 = s_{21}^2 + s_{33}^2$

h) $X_2 = u_1 w_1 + u_2 w_2$

i) Yes, the rank-two approximation lies in a plane. The plane captures the dominant components of the data.

j) $E = u_3 S_{3,3} V_3^T$, $\|E\|_F^2 = S_3^2$

k) rank-1: $\|E\|_F^2 = 626.70$

rank-2: $\|E\|_F^2 = 152.95$

$\|E\|_F^2$ using rank-2 approximation is less than that when using rank-1

3. $\hat{w}_\lambda = (X^TX - \lambda I)^{-1} X^T y = (V\Sigma u^T u \Sigma V^T - \lambda I)^{-1} V \Sigma u^T y$

$= (V\Sigma^2 V^T - \lambda I)^{-1} V \Sigma u^T y$.

(Since we can express λI as $V\lambda I V^T$ $(VV^T = I)$)

$= (V\Sigma^2 V^T - V\lambda I V^T) V \Sigma u^T y = V(\Sigma^2 - \lambda I)^{-1} V^T V \Sigma u^T y$

$= V(\Sigma^2 - \lambda I)^{-1} \Sigma u^T y$

So $S = (\Sigma^2 - \lambda I)^{-1} \Sigma$

Assignment 6

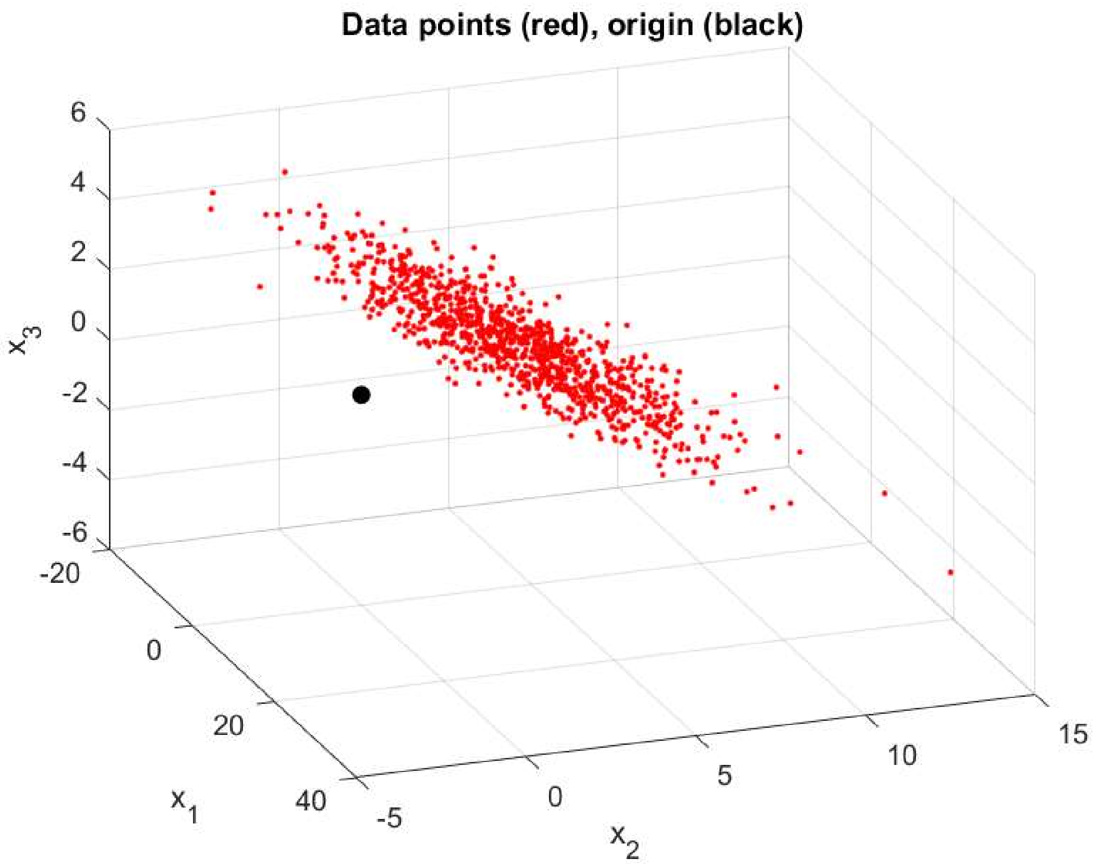
Contents

Prepare workspace

```
close all
clear
X = csvread('sdata.csv');
```

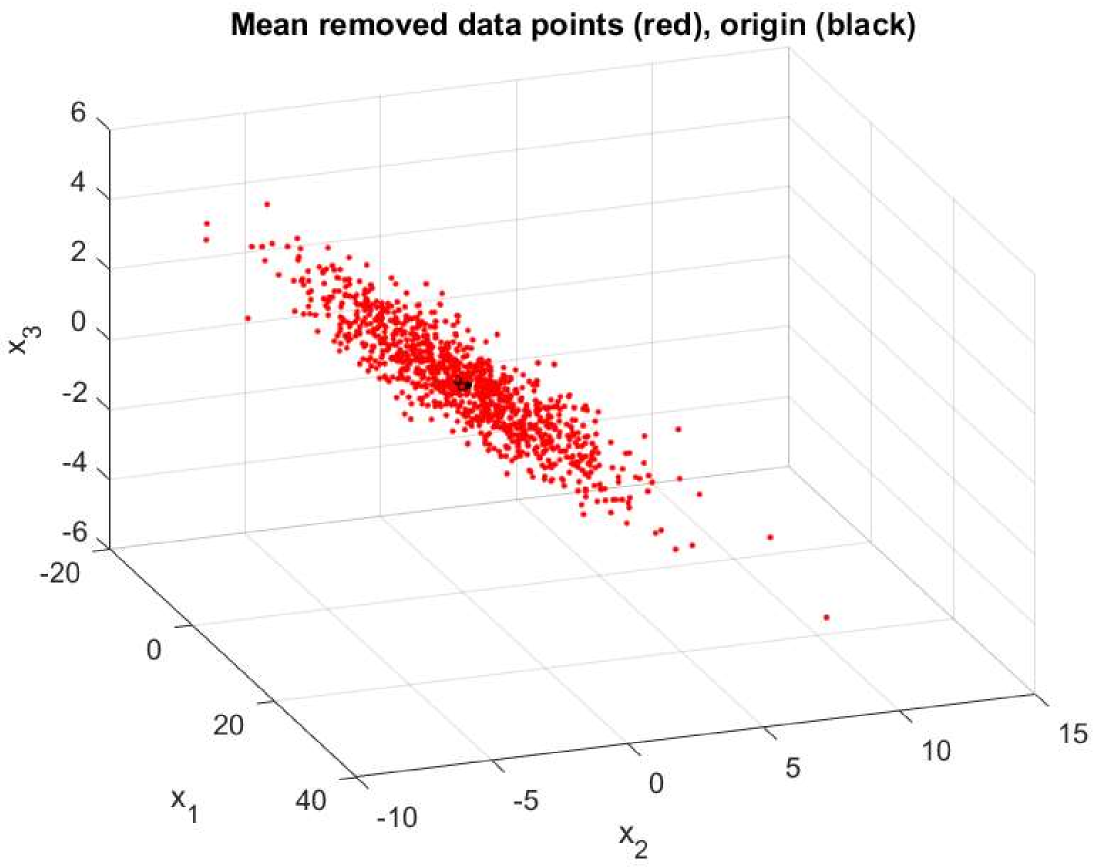
Display data

```
% Use rotate tool in the figure to view data from different angles
figure
scatter3( X(:,1), X(:,2), X(:,3), 'r.', 'LineWidth', 3 )
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')

hold on
scatter3(0,0,0,'MarkerEdgeColor','k',...
        'MarkerFaceColor','k')
hold off
title('Data points (red), origin (black)')
view(70,30)
```



Remove mean

```
mn = mean(X);

Xz = X - ones(1000,1)*mn;

figure
scatter3( Xz(:,1), Xz(:,2), Xz(:,3), 'r.', 'LineWidth', 3 )
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')

hold on
scatter3(0,0,0,'MarkerEdgeColor','k',...
        'MarkerFaceColor','k')
hold off
title('Mean removed data points (red), origin (black)')
view(70,30)
```



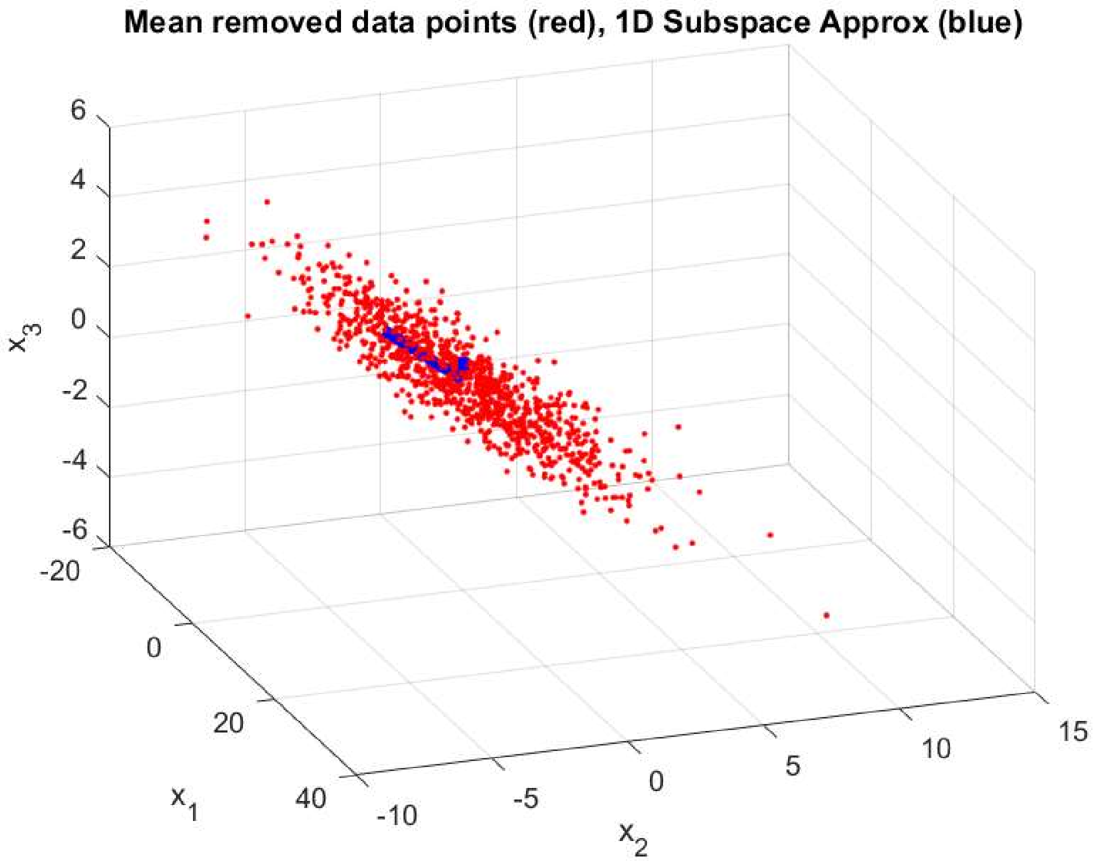
Take SVD to find best line

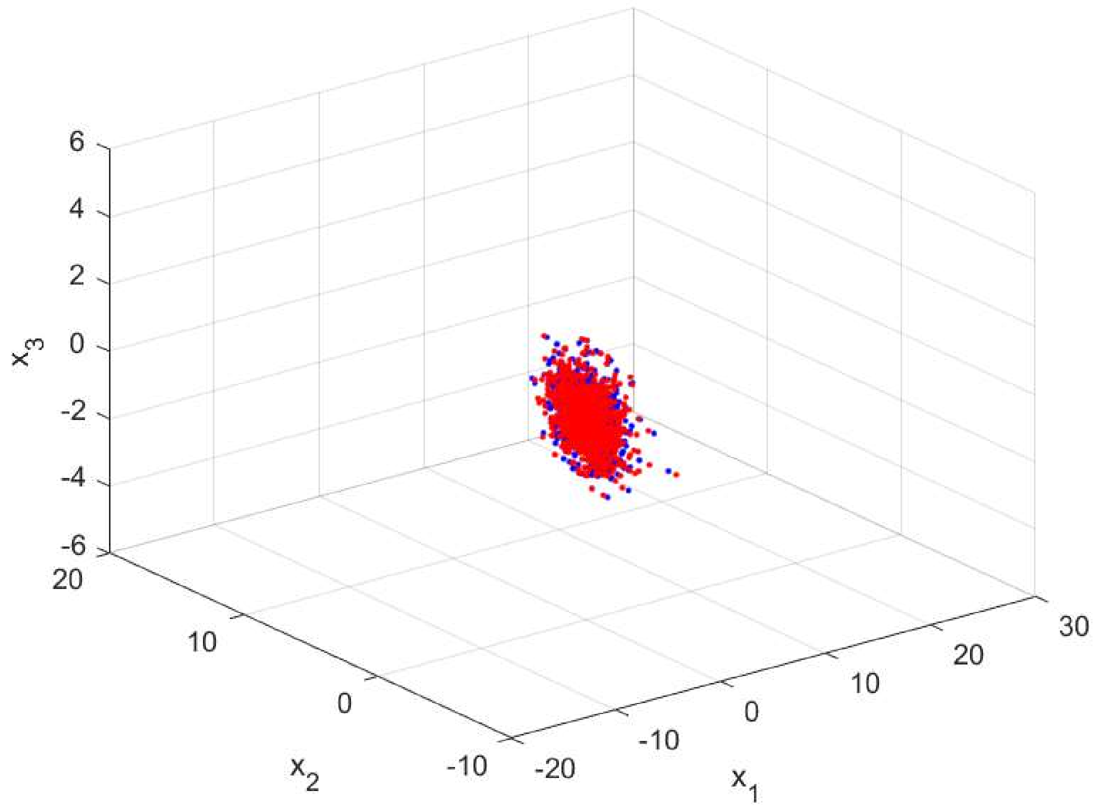
```
[U,S,V] = svd(Xz,'econ');

 a = V(:,1);  % Complete this line
```

Display best line on scatterplot

```
figure
scatter3( Xz(:,1), Xz(:,2), Xz(:,3), 'r.', 'LineWidth', 3 )
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')

a2 = V(:,2);
title('Mean removed data points (red), 1D Subspace Approx (blue)')

% Scale length of line by root-mean-square of data for display
scale1 = S(1,1)/sqrt(size(Xz,1));
scale2 = S(2,2)/sqrt(size(Xz,1));

hold on

plot3(scale1*[0;a(1)],scale1*[0;a(2)],scale1*[0;a(3)], 'b', 'LineWidth', 4)
plot3(scale2*[0;a2(1)],scale2*[0;a2(2)],scale2*[0;a2(3)], 'b', 'LineWidth', 4)
hold off

view(70,30)

Xz_2 = U(:,1)*S(1,1)*transpose(a) + U(:,2)*S(2,2)*transpose(a2);
figure
scatter3( Xz(:,1), Xz(:,2), Xz(:,3), 'r.', 'LineWidth', 3 )
```

```
hold on
scatter3( Xz_2(:,1), Xz_2(:,2), Xz_2(:,3), 'b.', 'LineWidth', 3 )
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')
hold off
```





Problem 3 part a

```
load('face_emotion_data.mat')
numRows = 0;
error = zeros(8,7);
for i=1:8
    heldoutx = X(numRows+1:numRows+16,:);
    heldouty = y(numRows+1:numRows+16,:);
    trainingx = X([1:numRows numRows+17:128],:);
    trainingy = y([1:numRows numRows+17:128],:);
    numRows = numRows+16;
    numRows_2 = 0;

    for j=1:7
        finaltrainingx = trainingx([1:numRows_2 numRows_2+17:112],:);
        finaltrainingy = trainingy([1:numRows_2 numRows_2+17:112],:);
        testx = trainingx(numRows_2+1:numRows_2+16,:);
        testy = trainingy(numRows_2+1:numRows_2+16,:);
        numRows_2 = numRows_2+16;

        [U,S,V] = svd(finaltrainingx);
        S_inverse = zeros(size(S));

        error_1 = zeros(1,9);
        error_2 = zeros(1,9);

        for k=1:9
            for r=1:k
                S_inverse(r,r) = 1/S(r,r);
            end
            w = V*transpose(S_inverse)*transpose(U)*finaltrainingy;
            y_predict_1 = sign(testx*w);
            error_1(1,k)=sum(y_predict_1 ~= testy)/16;
```

```
            y_predict_2 = sign(heldoutx*w);
            error_2(1,k)=sum(y_predict_2 ~= heldouty)/16;
        end
        [e_min,k_min] = min(error_1);
        error(i,j) = error_2(1,k_min);
    end
end
soln = mean(mean(error));
display(soln);
```

```
soln =

    0.1116
```

partb 3

```
lambda=[0;0.5;1;2;4;8;16];
error = zeros(8,7);

numRows=0;
for i=1:8
    heldoutx = X(numRows+1:numRows+16,:);
    heldouty = y(numRows+1:numRows+16,:);
    trainingx = X([1:numRows numRows+17:128],:);
    trainingy = y([1:numRows numRows+17:128],:);
    numRows = numRows+16;
    numRows_2 = 0;

    for j=1:7
        finaltrainingx = trainingx([1:numRows_2 numRows_2+17:112],:);
        finaltrainingy = trainingy([1:numRows_2 numRows_2+17:112],:);
        testx = trainingx(numRows_2+1:numRows_2+16,:);
        testy = trainingy(numRows_2+1:numRows_2+16,:);
        numRows_2 = numRows_2+16;

        [U,S,V] = svd(finaltrainingx);
        S_inverse = zeros(size(S));

        error_1 = zeros(1,9);
        error_2 = zeros(1,9);

        for k=1:7
            for r=1:9
                S_inverse(r,r) = S(r,r)/(S(r,r)^2+lambda(k));
            end
            w = V*transpose(S_inverse)*transpose(U)*finaltrainingy;
            y_predict_1 = sign(testx*w);
            error_1(1,k)=sum(y_predict_1 ~= testy)/16;
            y_predict_2 = sign(heldoutx*w);
            error_2(1,k)=sum(y_predict_2 ~= heldouty)/16;
        end
        [e_min,k_min] = min(error_1);
        error(i,j) = error_2(1,k_min);
    end
end
soln1 = mean(mean(error));
display(soln1);
```

```
soln1 =

    0.0223
```